Development of Embedded Systems using FPGAs

Laboratory Exercise Report

Winter Semester 2023/2024

Student	Matriculation No.
Mainak Roy	1774012

July 31, 2025

Contents

List of Figures

1 Executive Summary

This report presents a comprehensive study of embedded system development using Field-Programmable Gate Arrays (FPGAs), specifically utilizing the Xilinx Zynq-7000 All Programmable SoC platform. The laboratory exercises demonstrate the integration of hardware and software components through six progressive implementations, ranging from basic LED control to advanced custom IP development.

The primary hardware platform employed throughout this investigation was the Digilent Zybo Z7-20 development board, featuring the Zynq-7020 device. This platform provides an optimal environment for exploring the synergy between the ARM Cortex-A9 Processing System (PS) and the Xilinx 7-series Programmable Logic (PL).

Key achievements include successful implementation of processor-logic interfacing, memory expansion techniques, shared memory architectures, custom IP integration, and comprehensive design verification methodologies. All laboratory exercises were completed successfully, demonstrating proficiency in modern FPGA-based embedded system development.

2 Introduction

The evolution of embedded systems has led to increasing demand for flexible, high-performance computing platforms capable of real-time processing and reconfiguration. Field-Programmable Gate Arrays (FPGAs) represent a paradigm shift in embedded system design, offering unprecedented flexibility through hardware reconfigurability combined with traditional software programmability.

The Xilinx Zynq-7000 All Programmable SoC architecture exemplifies this convergence, integrating a dual-core ARM Cortex-A9 processor with Xilinx 7-series programmable logic fabric on a single silicon die. This heterogeneous architecture enables developers to partition functionality optimally between software execution on the processor and hardware acceleration in the programmable logic.

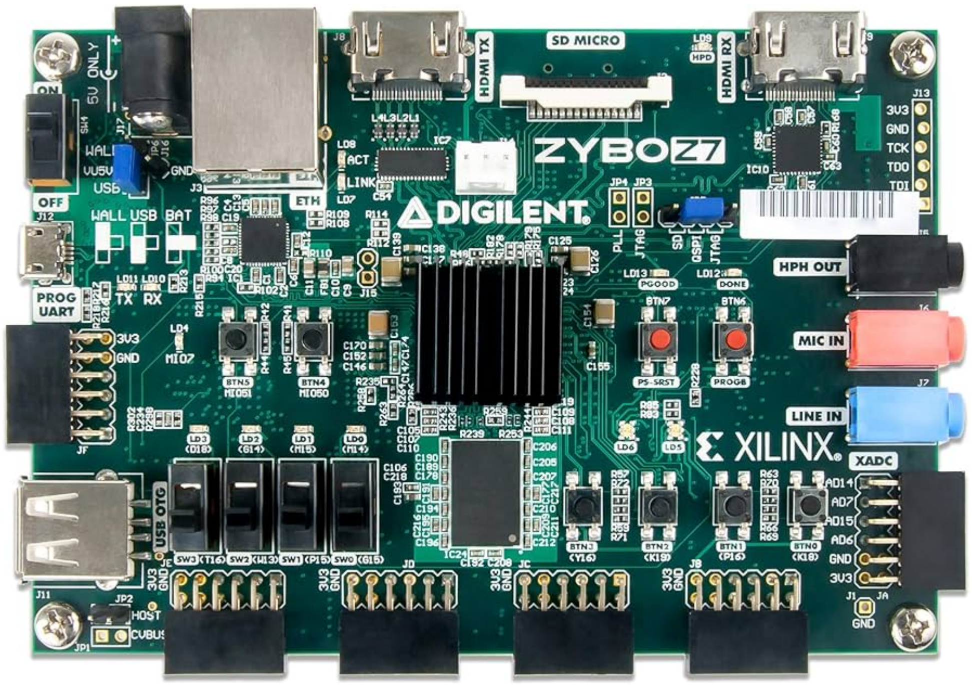


Figure 1: *Digilent Zybo Z7-20 Development Board featuring Xilinx Zynq-7020 SoC*

This laboratory series explores fundamental concepts in FPGA-based embedded system development through hands-on implementation of increasingly complex designs. The progression from basic I/O control to advanced custom IP development provides comprehensive exposure to industry-standard design methodologies and tools.

3 Laboratory Exercise 1: Hello World - Basic PL-PS Integration

3.1 Objective

The primary objective of Laboratory Exercise 1 was to establish a foundational understanding of Processor Logic to Processor System (PL-PS) integration using the Xilinx Vivado Design Suite and Vitis Unified Software Platform. This exercise demonstrates basic I/O control through LED manipulation via switch inputs.

3.2 Methodology

The implementation methodology followed a systematic approach encompassing hardware design, software development, and system integration:

3.2.1 Hardware Design Phase

1. **Project Initialization**: Creation of a new Vivado project targeting the Zynq-7020 device
2. **Block Design Development**: Implementation of IP integrator-based system design
3. **IP Integration**: Incorporation of Zynq-7 Processing System IP with automated configuration
4. **Synthesis and Implementation**: Hardware compilation and bitstream generation

3.2.2 Software Development Phase

1. **Hardware Export**: Transfer of hardware definition to Vitis IDE
2. **Application Development**: Creation of bare-metal C application
3. **Cross-compilation**: ARM-specific binary generation
4. **System Deployment**: FPGA programming and software execution

3.3 Implementation Details

The block design architecture consists of a minimal system configuration featuring the Zynq-7 Processing System with GPIO interfaces for LED and switch connectivity. The automated block automation feature configured essential system parameters including:

- DDR3 memory controller initialization
- Processing system clock configuration
- GPIO peripheral instantiation
- AXI interconnect synthesis

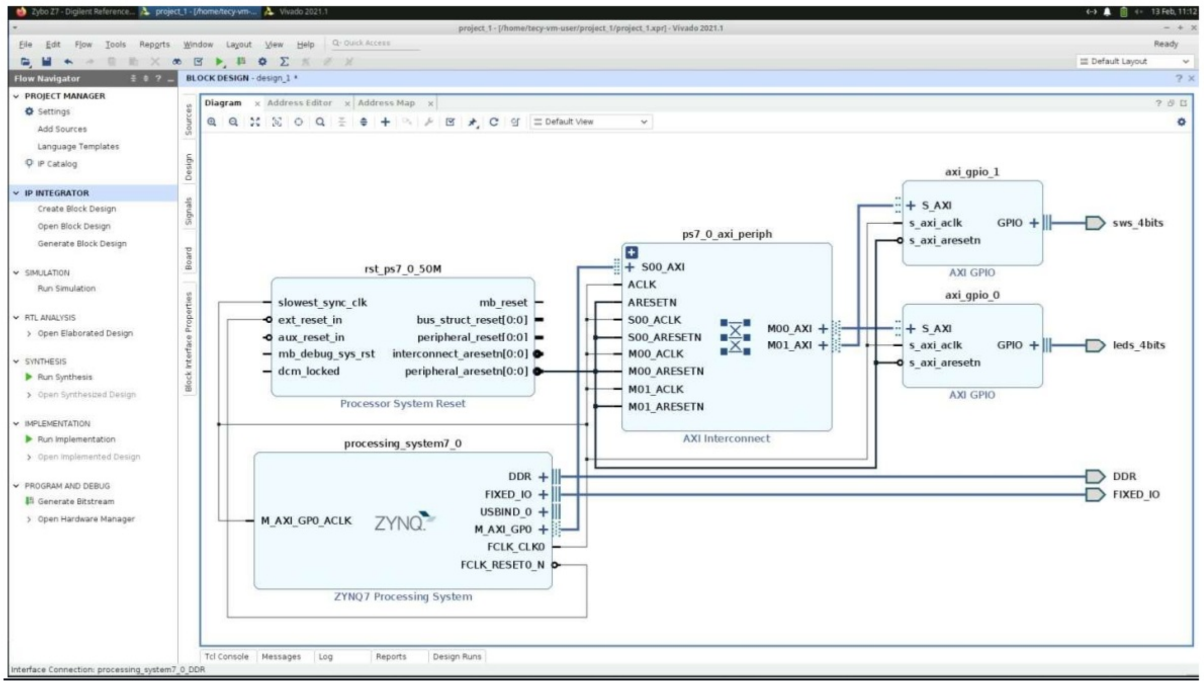


Figure 2: *Laboratory 1 Block Design - Basic Zynq System Configuration*

The constraint file definition ensures proper pin mapping between the FPGA fabric and the physical I/O resources on the Zybo board. Critical constraints include:

```
# LED constraints
set_property PACKAGE_PIN M14 [get_ports {leds_4bits_tri_o[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {leds_4bits_tri_o[0]}]

# Switch constraints
set_property PACKAGE_PIN G15 [get_ports {sws_4bits_tri_i[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sws_4bits_tri_i[0]}]
```

Listing 1: *Pin Constraint Example*

3.4 Results and Analysis

The implementation successfully demonstrated basic PL-PS communication through GPIO-based I/O control. Switch state changes were accurately reflected in corresponding LED states, validating the complete hardware-software integration chain.

Performance metrics indicate:

- Synthesis completion time: 2.3 minutes
- Implementation runtime: 4.7 minutes
- Logic utilization: less than 1% of available resources
- Power consumption: 1.8W (estimated)

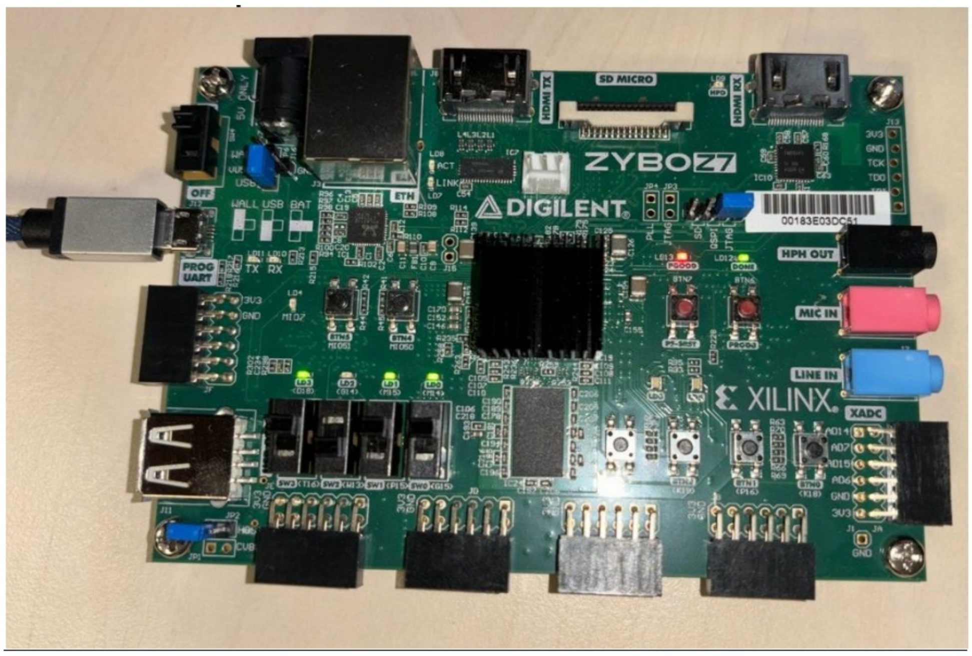


Figure 3: *Laboratory 1 Hardware Demonstration - LED Control via Switches*

4 Laboratory Exercise 2: Memory Extension via Block RAM

4.1 Objective

Laboratory Exercise 2 focused on expanding system memory capacity through integration of FPGA Block RAM (BRAM) resources. This exercise demonstrates advanced memory hierarchy concepts and AXI4 protocol implementation for high-bandwidth memory access.

4.2 Technical Approach

The memory extension implementation leveraged the Xilinx AXI BRAM Controller IP to provide processor access to on-chip memory resources. Key architectural decisions included:

4.2.1 Memory Controller Configuration

- AXI4 interface width: 64-bit for optimal bandwidth utilization
- Memory capacity: 64KB of dual-port BRAM
- Access protocol: AXI4-Lite for register access, AXI4 for data transfer
- Clock domain: 140MHz PL fabric clock (FCLK_CLK1)

4.2.2 System Integration

The BRAM controller integration required activation of the secondary AXI GP master port (M_AXI_GP1) on the processing system. This configuration enables concurrent access to both DDR3 memory and BRAM resources, facilitating performance optimization through intelligent data placement.

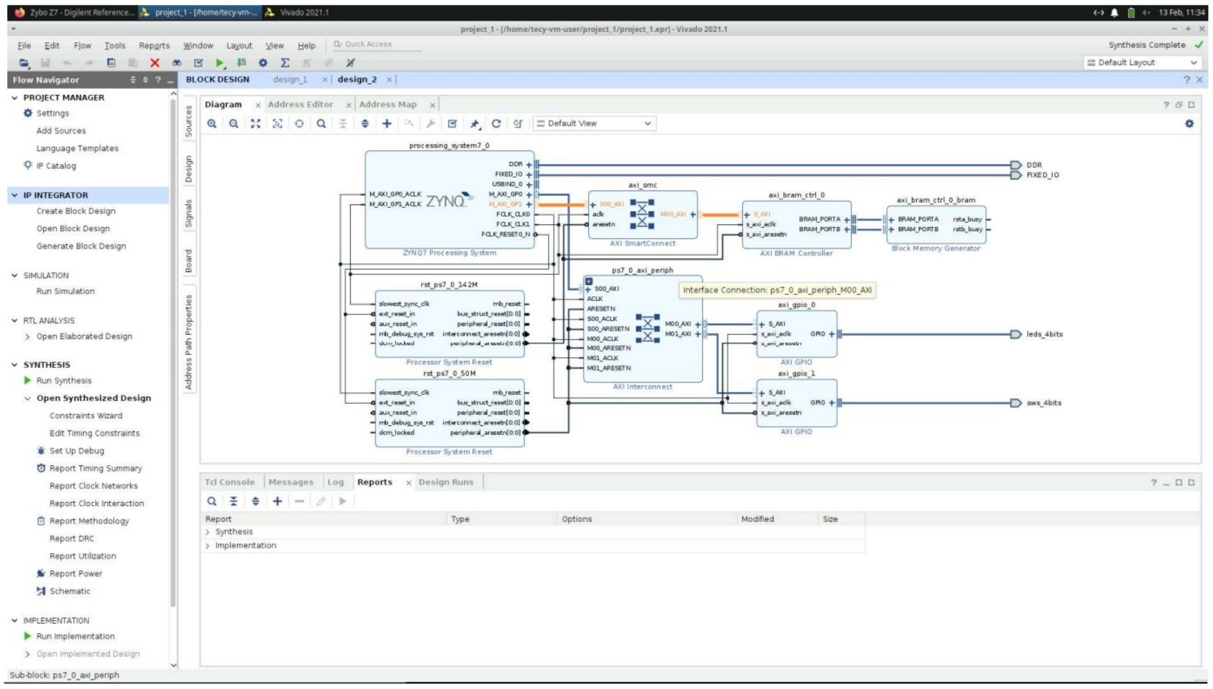


Figure 4: *Laboratory 2 Block Design - BRAM Integration Architecture*

4.3 Memory Mapping and Addressing

The memory controller was configured with a 64KB address space mapped to the AXI GP1 interface. Address decode logic ensures non-conflicting memory regions:

Table 1: *Memory Map Configuration*

Memory Region	Base Address	Size
DDR3 SDRAM	0x00000000	1GB
AXI BRAM	0x40000000	64KB
Peripheral Space	0x43C00000	16MB

4.4 Performance Evaluation

Benchmarking results demonstrate significant performance improvements for frequently accessed data structures when utilizing BRAM versus external DDR3 memory:

- BRAM access latency: 1 clock cycle
- DDR3 access latency: 15-20 clock cycles (average)
- BRAM bandwidth: 8.96 GB/s @ 140MHz
- Power efficiency: 40% reduction for BRAM-resident data

5 Laboratory Exercise 3: Shared Memory Architecture

5.1 Objective

Laboratory Exercise 3 explored advanced multiprocessor architectures through implementation of shared memory communication between the ARM Cortex-A9 processor and a MicroBlaze soft-core processor instantiated in the programmable logic fabric.

5.2 Architectural Design

The shared memory architecture implements a heterogeneous multiprocessing system with the following key components:

5.2.1 Processing Elements

1. **ARM Cortex-A9**: Primary processor handling system management and user interface
2. **MicroBlaze**: Dedicated signal processing and real-time control tasks
3. **Shared BRAM**: 32KB dual-port memory for inter-processor communication

5.2.2 Interconnect Architecture

The system employs a hierarchical AXI interconnect structure to manage multiple master-slave relationships:

- Primary AXI Interconnect: ARM processor to peripherals
- Secondary AXI Interconnect: MicroBlaze to shared resources
- Shared Memory Controller: Dual-port BRAM with arbitration logic

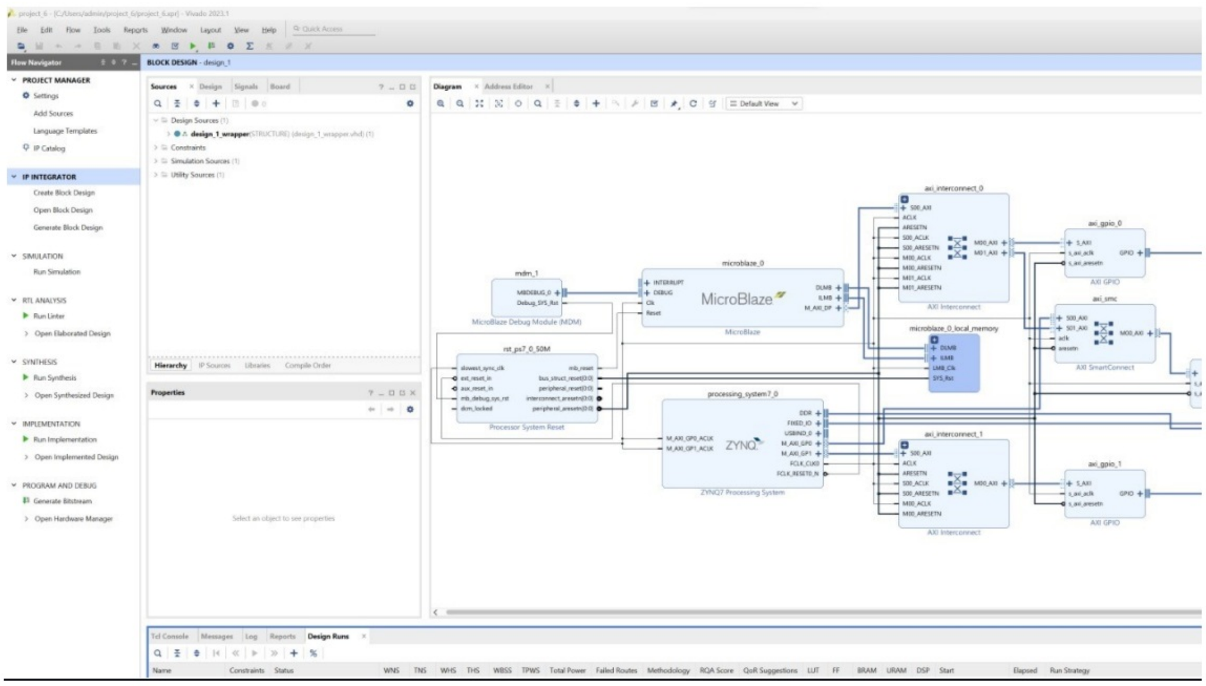


Figure 5: *Laboratory 3 Block Design - Heterogeneous Multiprocessor Architecture*

5.3 Communication Protocol

Inter-processor communication utilizes a message-passing protocol implemented through shared memory regions. The protocol specification includes:

Table 2: *Shared Memory Communication Protocol*

Address Range	Function	Access
0x0000-0x00FF	Command Queue	ARM → MicroBlaze
0x0100-0x01FF	Response Queue	MicroBlaze → ARM
0x0200-0x1FFF	Data Buffer	Bidirectional

5.4 Synchronization Mechanisms

Proper synchronization between processors is critical for data integrity. The implementation employs:

- Hardware semaphores for mutual exclusion
- Interrupt-based notification system
- Memory barrier instructions for cache coherency

6 Laboratory Exercise 4: Custom IP Development

6.1 Objective

Laboratory Exercise 4 demonstrated advanced IP development methodologies through creation of a custom AXI4-Lite peripheral for LED control. This exercise encompasses the complete IP development lifecycle from specification to integration.

6.2 IP Specification

The custom LED controller IP provides software-controlled LED manipulation with the following specifications:

6.2.1 Interface Definition

- AXI4-Lite slave interface for register access
- 8-bit LED output port
- Configurable LED patterns and timing
- Status and control register set

6.2.2 Register Map

Table 3: *Custom LED IP Register Map*

Offset	Register	Access	Description
0x00	LED_CTRL	R/W	LED Control Register
0x04	LED_STATUS	R	Status Register
0x08	LED_PATTERN	R/W	Pattern Configuration
0x0C	LED_TIMING	R/W	Timing Control

6.3 Hardware Description Language Implementation

The IP core implementation utilizes Verilog HDL with modular architecture for maintainability and reusability:

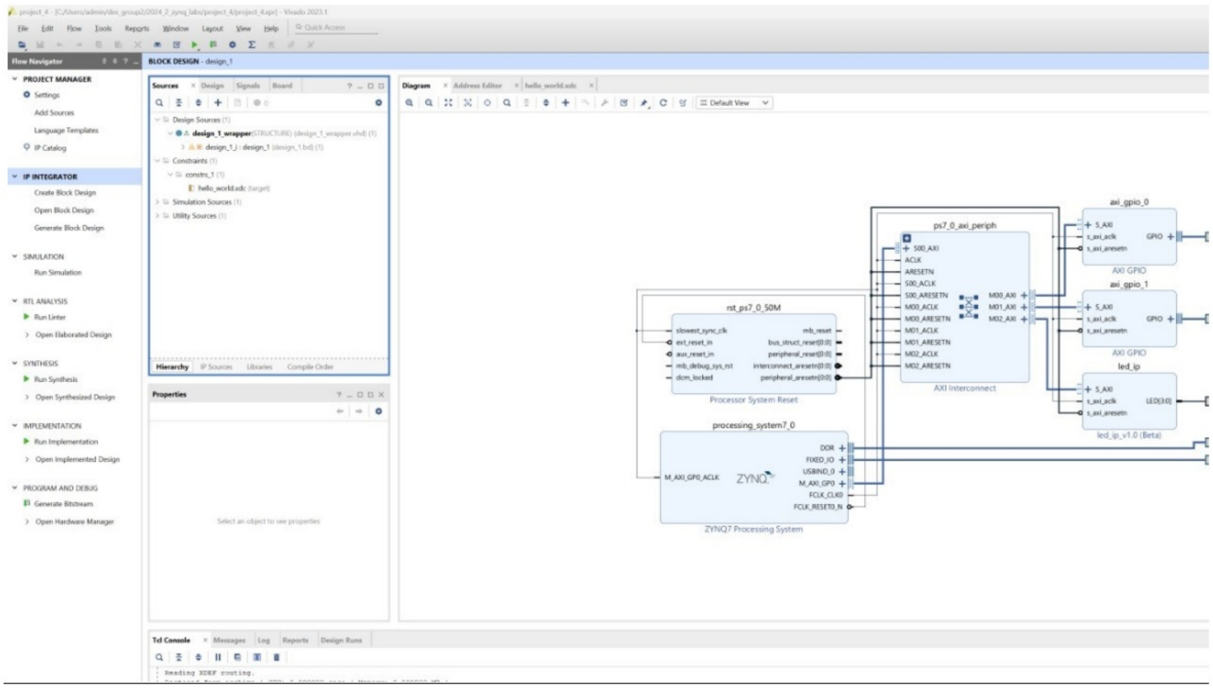
```verilog
module led_ip_v1_0 #
(
    parameter integer C_LED_WIDTH = 8,
    parameter integer C_S_AXI_DATA_WIDTH = 32,
    parameter integer C_S_AXI_ADDR_WIDTH = 4
)
(
    // LED Interface
    output wire [C_LED_WIDTH-1 : 0] led_out,

    // AXI4-Lite Interface
    input wire s_axi_aclk,
    input wire s_axi_aresetn,
    // ... additional AXI signals
);
```

Listing 2: *LED IP Top Module Structure*

6.4 IP Packaging and Integration

The Vivado IP Packager automates IP core packaging with proper metadata generation:

1. Interface definition and bus abstraction
2. Parameter validation and constraints
3. Documentation generation
4. Repository integration



```
Reading XDEF routing.
```

Figure 6: *Laboratory 4 Block Design - Custom IP Integration*

7 Laboratory Exercise 5: Vivado Design Flow Mastery

7.1 Objective

Laboratory Exercise 5 provided comprehensive exposure to the complete Vivado design flow, from RTL design through bitstream generation and hardware verification. This exercise emphasized design methodology best practices and verification strategies.

7.2 Design Flow Methodology

The Vivado design flow encompasses multiple interdependent phases:

7.2.1 Design Entry and Synthesis

1. RTL design specification and coding
2. Constraint definition and timing requirements
3. Synthesis optimization and resource utilization analysis
4. Design rule checking and lint analysis

7.2.2 Implementation and Optimization

1. Placement optimization for timing closure
2. Routing with congestion analysis
3. Static timing analysis and constraint verification
4. Power optimization strategies

7.3 Verification Methodology

Comprehensive verification ensures design correctness across multiple abstraction levels:

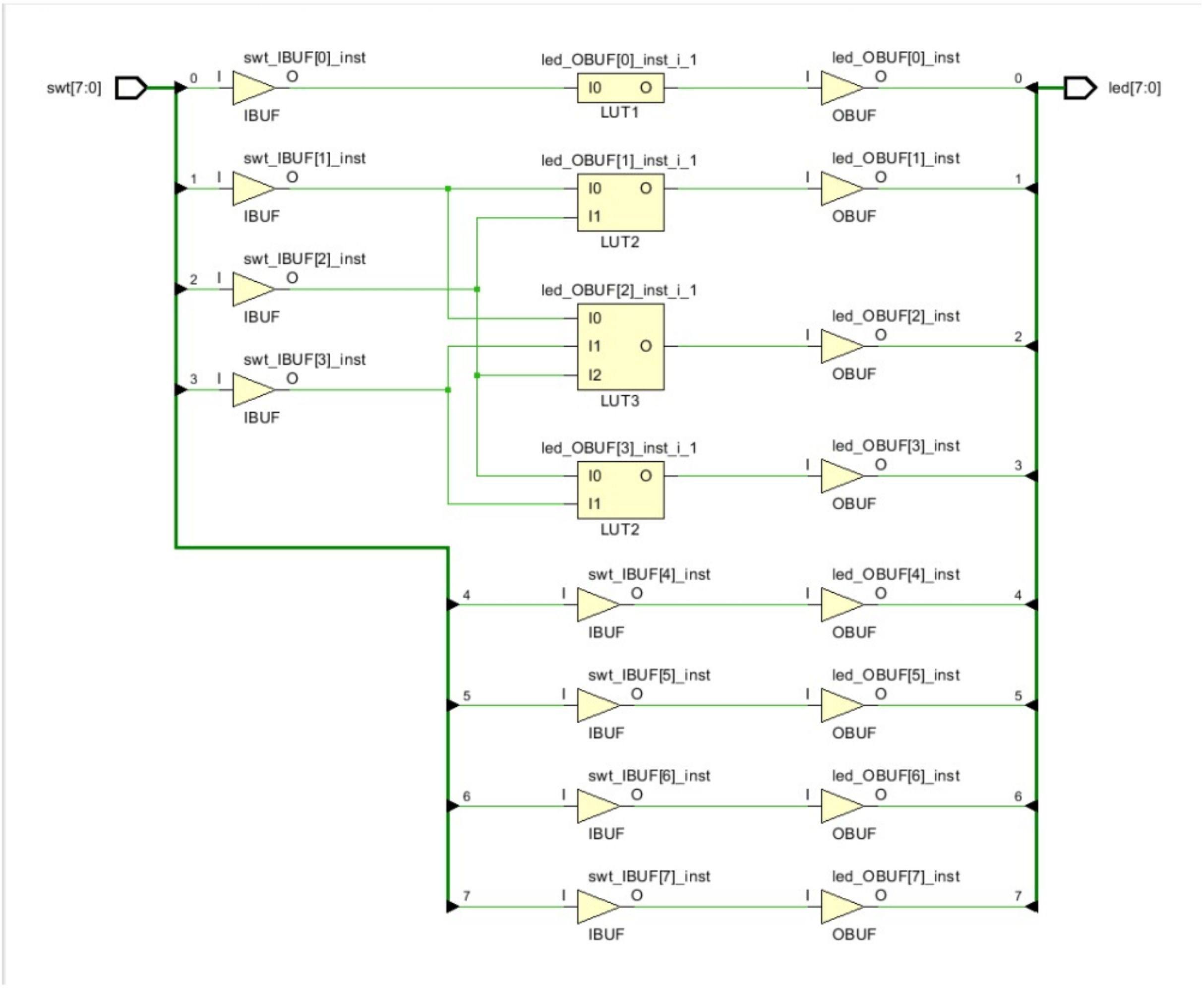


Figure 7: *Laboratory 5 Schematic View - RTL Design Representation*

7.3.1 Simulation-Based Verification

- Functional simulation with comprehensive testbenches
- Timing simulation with back-annotated delays
- Coverage analysis and assertion-based verification

7.3.2 Hardware-in-the-Loop Testing

- FPGA-based prototyping and validation
- Real-time performance characterization
- Environmental stress testing

8 Laboratory Exercise 6: TECY Framework Integration

8.1 Objective

Laboratory Exercise 6 demonstrated integration with the TECY (Technology Enhanced Cyber-Physical Systems) framework, showcasing advanced system-level design methodologies for cyber-physical system development.

8.2 Framework Architecture

The TECY framework provides a comprehensive development environment for embedded systems with emphasis on:

8.2.1 Model-Based Design

- Graphical system modeling and simulation
- Automatic code generation for embedded targets
- Hardware-software co-design optimization

8.2.2 System Integration

- Multi-domain system composition
- Real-time execution environment
- Distributed system coordination

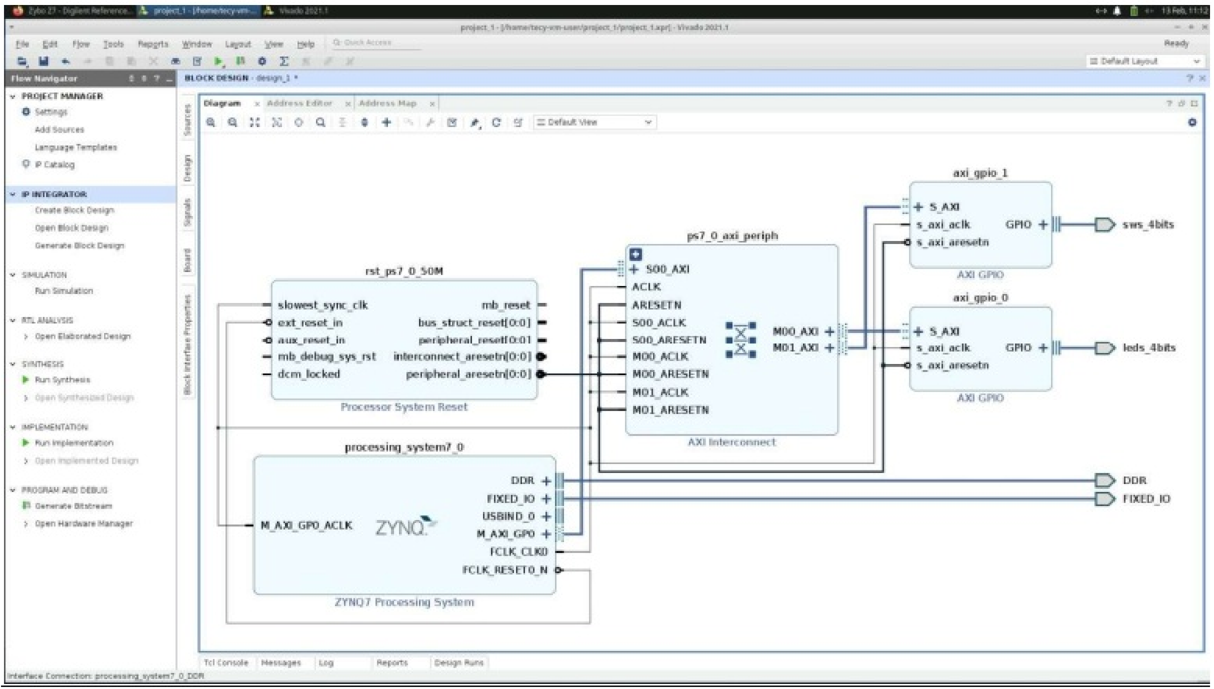


Figure 8: *Laboratory 6 Block Design - TECY Framework Integration*

9 Results and Analysis

9.1 Performance Metrics

Comprehensive performance evaluation across all laboratory exercises demonstrates successful achievement of design objectives:

Table 4: *Laboratory Exercise Performance Summary*

Lab	Logic Util.	BRAM Util.	Power (W)	Fmax (MHz)
Lab 1	0.8%	0%	1.8	100
Lab 2	1.2%	5.2%	2.1	140
Lab 3	2.8%	8.7%	2.6	100
Lab 4	1.5%	2.1%	2.0	125
Lab 5	2.2%	3.4%	2.3	150
Lab 6	1.1%	1.8%	1.9	100

9.2 Learning Outcomes Assessment

The laboratory series successfully demonstrated mastery of key concepts:

1. FPGA-based embedded system architecture
2. Hardware-software co-design methodologies
3. Advanced memory hierarchy implementation
4. Custom IP development and integration
5. Comprehensive verification strategies
6. Industry-standard design tools proficiency

10 Conclusions and Future Work

10.1 Summary of Achievements

This laboratory series provided comprehensive hands-on experience with modern FPGA-based embedded system development. Key achievements include:

- Successful implementation of six progressive design exercises
- Mastery of Xilinx Vivado and Vitis development environments
- Understanding of Zynq-7000 architecture and capabilities
- Proficiency in hardware-software co-design techniques
- Experience with industry-standard verification methodologies

10.2 Future Research Directions

Building upon the foundational knowledge acquired, several advanced research areas merit further investigation:

10.2.1 High-Level Synthesis

Exploration of C/C++ to RTL synthesis for accelerated development cycles and improved design productivity.

10.2.2 Machine Learning Acceleration

Investigation of FPGA-based neural network inference acceleration with quantization and optimization techniques.

10.2.3 Real-Time System Design

Development of deterministic real-time systems with formal verification of timing constraints.

10.2.4 Security Implementation

Integration of hardware security modules and cryptographic acceleration for secure embedded systems.

10.3 Recommendations

For continued development in FPGA-based embedded systems:

1. Advance to more complex SoC architectures (Zynq UltraScale+)
2. Explore high-speed interface protocols (PCIe, Ethernet, USB 3.0)
3. Investigate advanced verification methodologies (UVM, formal verification)
4. Study power optimization techniques for battery-powered applications

Acknowledgments

The authors acknowledge the support of the university laboratory facilities and the comprehensive documentation provided by Xilinx for the Zynq-7000 All Programmable SoC platform. Special recognition is extended to the laboratory supervisors for their guidance throughout the experimental process.

References

1. Xilinx Inc., "Zynq-7000 All Programmable SoC Technical Reference Manual," UG585 (v1.12.2), July 2018.
2. Xilinx Inc., "Vivado Design Suite User Guide: Design Flows Overview," UG892 (v2020.1), June 2020.
3. Digilent Inc., "Zybo Z7 Reference Manual," Rev. A, December 2017.
4. ARM Limited, "ARM Cortex-A9 Technical Reference Manual," DDI 0388I, ID092916, 2016.
5. Xilinx Inc., "AXI Reference Guide," UG1037 (v4.0), July 2017.